

## Code Conversion

Numbers can be represented by a large variety of codes. The binary code is the most natural, the simplest, and the one most commonly used in high-speed computer systems. For convenience this code is often grouped in 3-bit groups and called an octal code. However, since this code is simply a different way of interpreting the binary code, all the features of the binary code are retained.

Unfortunately, a different numbering system based on the number 10 is in everyday use and mixed numbering systems are also used for some special applications (time, angles, etc.). This ambiguity has created a need for binary-to-BCD (binary coded decimal) and BCD-to-binary converter circuits.

The number of bits and digits involved, the time available and the amount of general purpose (perhaps even microprogrammed) logic available in the system are important factors in selecting one of the many different methods available for code conversion.

Any arbitrary code can be converted into any other arbitrary code by using a Read Only Memory (ROM) as a lookup table. This method is very fast with bipolar ROMs, but in most cases it is unnecessarily expensive, since most codes show some kind of regularity. Cost effective MSI circuits can take advantage of this regularity and provide a more economical solution.

Binary adders are used in high-speed parallel BCD-to-binary conversion. Every bit in a BCD number can be expressed as a binary number, and their sum is the binary equivalent of the whole BCD number. BCD adders such as the 'F583 are available for performing high-speed BCD arithmetic.

Converting a 2-digit BCD number into a 7-bit binary number is accomplished simply and economically with two 4-bit adders. The necessary interconnections are determined by first expressing each of the weighted BCD bits in terms of numbers that are powers of two.

$$80 = 64 + 16 = 2^6 + 2^4$$

$$40 = 32 + 8 = 2^5 + 2^3, \text{ etc.}$$

Arranging the BCD and binary numbers in an orderly array, as shown in Table 5-8, makes it easy to see which of the BCD inputs must be summed into the various binary outputs. For example, the  $2^0$  output is the least significant bit of the unit's BCD digit, while inputs 2 and 10 must be summed to produce the  $2^1$  output. Notice that the  $2^3$  sum has more than two inputs (8, 10 and 40) and therefore cannot be formed in a single adder stage. Thus, for the  $2^3$  output, the sum is partially formed in the first adder package and completed in the second, as shown in the connection diagram. Inputs marked with a T must be terminated HIGH for active LOW inputs.

Table 5-8

|                |       | BCD Inputs |   |   |   |    |    |    |    |
|----------------|-------|------------|---|---|---|----|----|----|----|
|                |       | 1          | 2 | 4 | 8 | 10 | 20 | 40 | 80 |
| Binary Outputs | $2^0$ | X          |   |   |   |    |    |    |    |
|                | $2^1$ |            | X |   |   | X  |    |    |    |
|                | $2^2$ |            |   | X |   |    | X  |    |    |
|                | $2^3$ |            |   |   | X | X  |    | X  |    |
|                | $2^4$ |            |   |   |   |    | X  |    | X  |
|                | $2^5$ |            |   |   |   |    |    | X  |    |
|                | $2^6$ |            |   |   |   |    |    |    | X  |

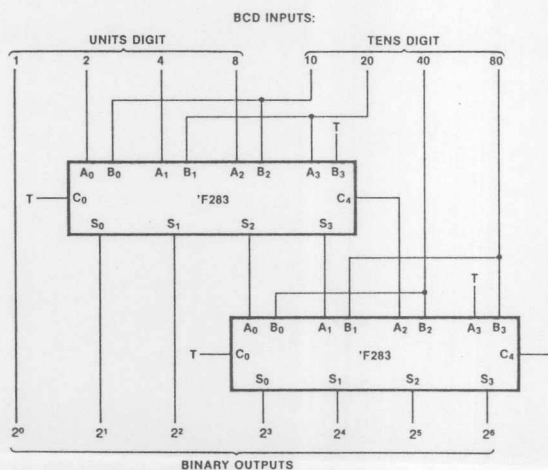
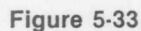


Figure 5-32 2-Digit BCD to 7-Bit Binary Converter

## 5



mode input control, M, would be tied HIGH for active LOW operation and tied LOW for active HIGH operation. Note that one of the Carry Ins is not connected to either the HIGH or LOW bus, but rather to the BCD 10. One of the Carry Outs can be ignored since it cannot be active for any legitimate input condition.

## Bit Serial Binary-to-BCD Converter

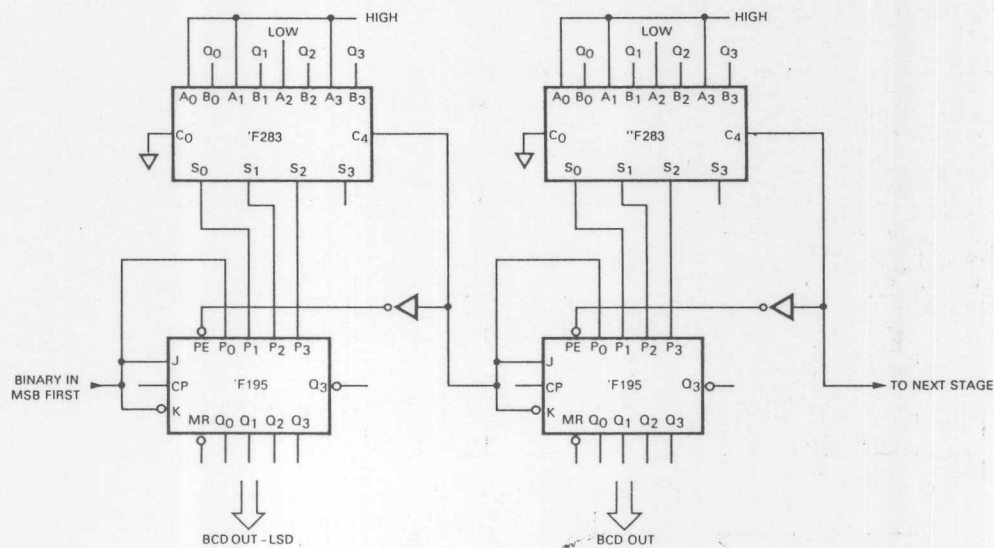


Figure 5-34

The reverse of the BCD-to-binary algorithm is used for binary-to-BCD conversion. The binary word is shifted, most significant bit first, into a shift register consisting of several series-connected 'F195. Each shift doubles the contents of the registers in terms of BCD notation. Therefore, a correction is required whenever any of the 4-bit registers contains a number greater than four, which when shifted generates a non-BCD code. This correction is performed by adding three to the contents

of the register and inserting the sum one bit downstream into the parallel data inputs. By adding 11 and then ignoring the most significant bit, the same 4-bit adder also detects whether or not the correction is necessary. A binary number is completely converted when its LSB has been shifted in. This circuit can be used for any number of bits and digits. It requires only one 'F195 4-bit shift register, one 'F283 4-bit adder, and one inverter for each resulting BCD digit.

## Serial-In, Serial-Out BCD-to-Binary Converter

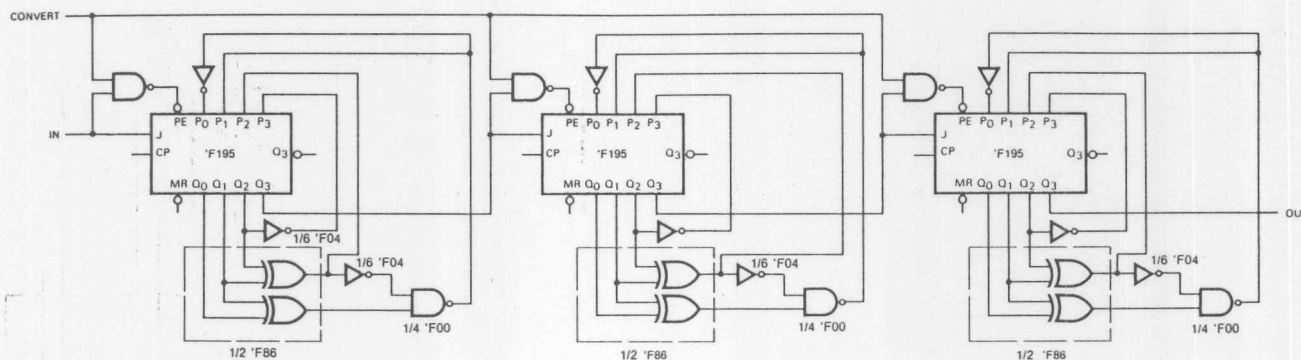


Figure 5-35

A well-known algorithm generates the binary equivalent of a BCD number by repeatedly dividing it by two. The series of least significant bits generated is the binary output, least significant first. This algorithm can be implemented with 'F195 shift registers and some gates or adders.

When a BCD number is stored in the 'F195 shift register with its LSB in the  $Q_3$  stage, a right shift effectively divides it by two. A problem arises if the LSB of the more significant digit is a one, implying a value of 10 with respect to the first digit. Shifting this one into the  $Q_0$  position changes the 10 to an 8, instead of dividing it by 2. To correct for this, a 3 must be subtracted from the new contents of the 'F195 register. The circuit shown provides a gate-minimized implementation of this

algorithm using the parallel inputs of the 'F195 for the correction. It converts a 4-digit (9999) BCD number into its 14-bit binary equivalent. Operation is started by bit-serially shifting in the three least significant BCD digits (LSB of the LSD first) while the Convert input is LOW. The actual conversion starts when the three digits have been shifted in and the LSB of the most significant digit is being applied to the serial input. At this point, the Convert input is made HIGH, activating the three correction networks whenever there is a one to be shifted into any of the registers. The next fourteen clock pulses shift out the binary result, LSB first. This circuit can be used for any number of digits. It requires only one 4-bit shift register with a conversion network for each decimal digit except the MSD.

## BCD-to-Binary Converter Using Counters

Counters can be used for binary-to-BCD or for BCD-to-binary conversion, if sufficient time is available. The code to be converted is loaded into a counter, which is then counted down while another counter is counted up. When the first counter reaches zero the second counter has reached the original numeric value represented in the desired code. This method is easily expanded and can also be used for mixed modulo codes (like 6/10 for minutes and seconds, 36/10 for degrees of angle). It is also very easy to perform accumulation.

Figure 5-36 is a BCD-to-binary converter capable of taking a 5-digit BCD number and converting it into a 17-bit binary number in less than 5ms using a clock generator running at 40MHz. The circuit, as shown, is completely self-contained, including a debounce/edge detect circuit for use with a push button to initiate the conversion. Only eleven integrated circuit packages are required.





## Binary Angle to BCD Converter With Display

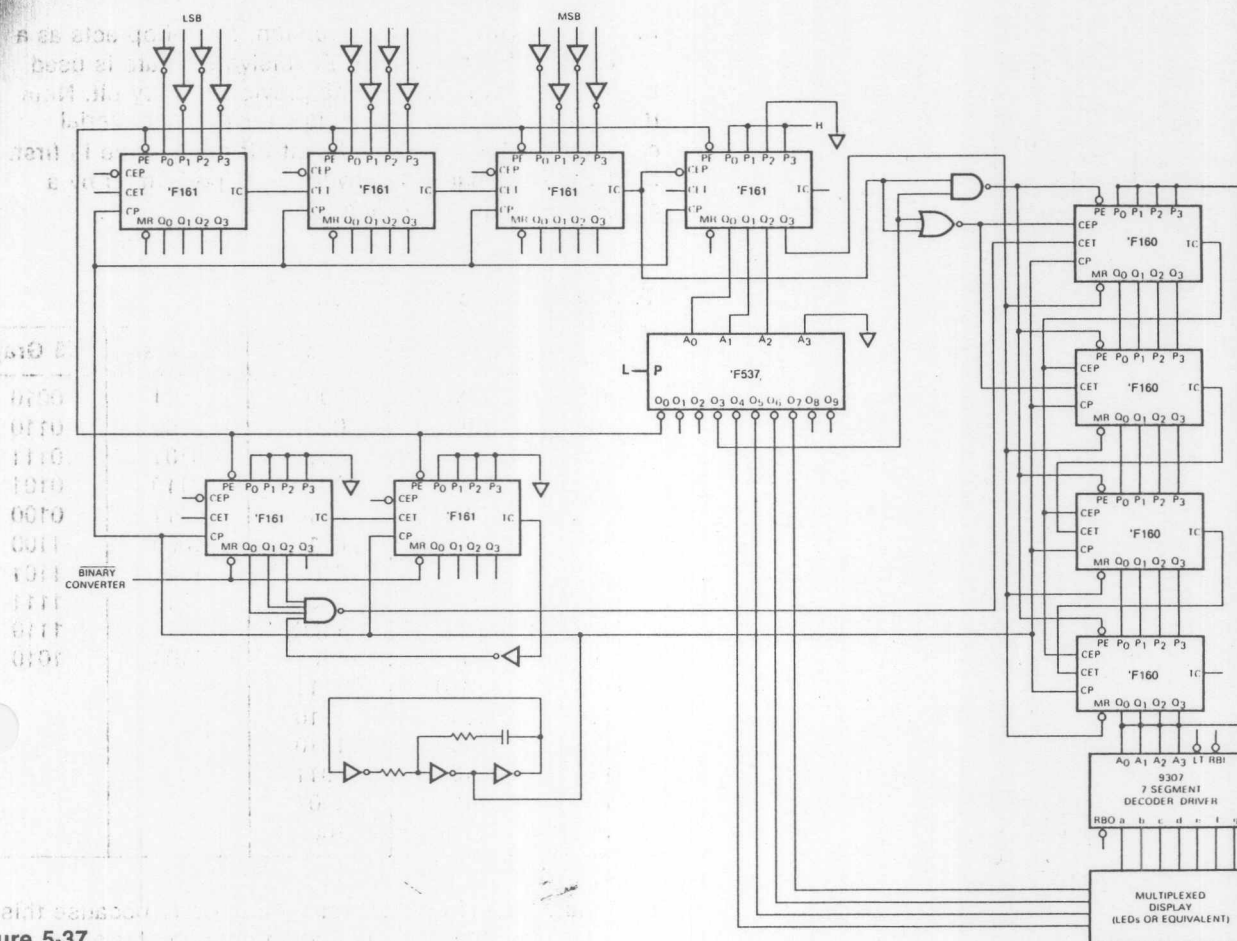


Figure 5-37

This converter can perform a code conversion and display the result continuously in binary coded decimal (BCD). The converter, as shown, operates in either of two modes, binary-to-BCD or binary angle-to-BCD. In the binary angle code, the most significant bit represents 180° (half circle), the next 90° (quarter circle), the third 45° (eighth circle), etc. The converter accepts twelve bits of this code and converts it to degrees and tenths of degrees (0.0° to 359.9°). The converter is a self-contained circuit requiring only seventeen packages capable of running at a 30MHz clock rate. The BCD output is displayed in an economical manner by multiplexing the four digits.

The circuit operates by automatically loading the complement of the input code into the binary counter at the beginning of the cycle while the decimal counter is cleared. The binary counter reaches terminal count after  $n$  clock pulses where  $n$  = binary input number. In the binary conversion mode, the decade counter also counts  $n$  times and thus reaches the BCD equivalent of the binary input. If the ability to convert angles to BCD is not needed, the two lower 'F161s and two gates are not needed.

After the conversion, the BCD data is displayed by multiplexing the digits, most significant digit first. The four stage binary counter shifts the digits through the decimal counter while enabling one display digit at a time by counting 4096 clock pulses per digit. After displaying the least significant digit, the cycle is repeated.

In the binary angle-to-BCD mode, the decimal counter is incremented at a slower rate than the binary counter to adjust for the weights of the binary angle bits entered in the binary counter. The most significant bit in a binary-to-BCD conversion has a weight of  $2^{11}$  or 2048, but in this binary angle-to-BCD conversion it has a weight of 180° or 1800 tenths of a degree. The BCD counter is therefore incremented at 1800/2048, or 225/256 the rate of the binary counter by inhibiting the decimal counter 31 times during every 256 clock pulses. This rate multiplier function is performed by two 'F161s (modulo 256 counter) and two gates decoding every eighth state, except TC of the counter, for a total of 31 states. These evenly distributed inhibit pulses minimize the conversion error.

## Gray Code Conversions

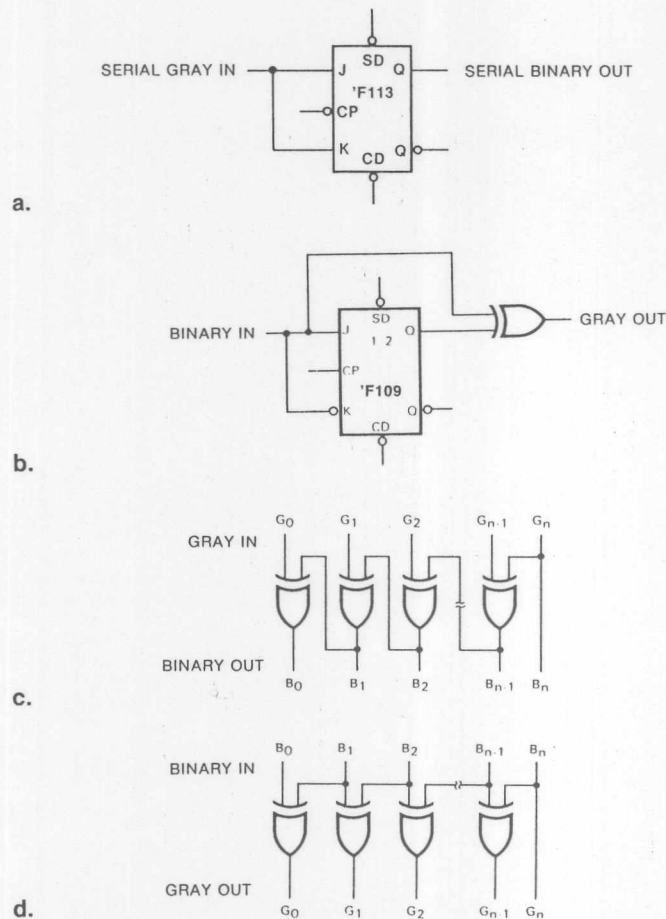


Figure 5-38

Binary codes are not particularly suited for electrical or electro-optical encoder systems (angular position shift encoders, etc.) because a movement from one state to the next often results in more than one bit change; i.e., from seven to eight, the binary code changes from 0111 to 1000. Such bit changes can never really be simultaneous, so the encoder always generates erroneous transient codes when switching between certain positions. This problem is avoided with a Gray code because only one bit changes between adjacent states. The Gray code is a non-weighted code and awkward for other applications. It must be converted to binary or BCD before any arithmetic can be performed.

In Gray-to-binary serial conversion, a flip-flop that toggles for every logic one performs the conversion. The most significant bit, however, must come in first. Gray-to-binary parallel conversion is performed by a series of Exclusive-OR gates.

In binary-to-Gray serial conversion, a flip-flop acts as a 1-bit delay element and an Exclusive-OR gate is used between the present and the previous binary bit. Note that in this case, as well as in Gray-to-binary serial conversion, the most significant bit must come in first. Binary-to-Gray parallel conversion is performed by a series of Exclusive-OR gates.

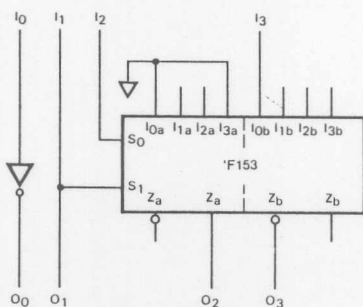
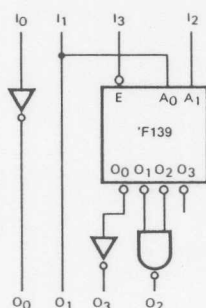
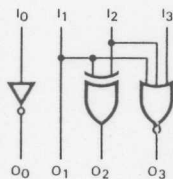
Table 5-9 Excess 3 Gray Code

| Decimal | Binary | Gray | X3 Binary | X3 Gray |
|---------|--------|------|-----------|---------|
| 0       | 0000   | 0000 | 0011      | 0010    |
| 1       | 0001   | 0001 | 0100      | 0110    |
| 2       | 0010   | 0011 | 0101      | 0111    |
| 3       | 0011   | 0010 | 0110      | 0101    |
| 4       | 0100   | 0110 | 0111      | 0100    |
| 5       | 0101   | 0111 | 1000      | 1100    |
| 6       | 0110   | 0101 | 1001      | 1101    |
| 7       | 0111   | 0100 | 1010      | 1111    |
| 8       | 1000   | 1100 | 1011      | 1110    |
| 9       | 1001   | 1101 | 1100      | 1010    |
| 10      | 1010   | 1111 |           |         |
| 11      | 1011   | 1110 |           |         |
| 12      | 1100   | 1010 |           |         |
| 13      | 1101   | 1011 |           |         |
| 14      | 1110   | 1001 |           |         |
| 15      | 1111   | 1000 |           |         |

Decimal systems use Excess 3 Gray Code because this code has the feature of changing only one bit at a time even on a nine-to-zero transition. Excess 3 Gray Code is detected or generated in the same manner as Gray codes, except adding a three to the binary value for binary-to-Excess 3 conversion and subtracting a three (i.e., adding binary 13) from the binary value for Excess 3-to-binary conversion.

## Generating Nines Complements

a.



c.

Figure 5-39

The one complement of a binary number is easily generated by inverting each bit. The equivalent in a decimal (BCD) system, nines complement, is not that easy. These three circuits convert a 1-digit BCD input into its nines complement. They use about one equivalent gate or MSI package per digit (decade).

## Controlled Nines Complement Circuit Using Two Gate Packages

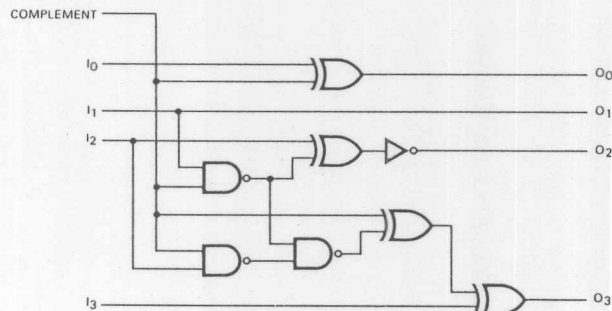


Figure 5-40

Table 5-10 Truth Table

| I <sub>0</sub> | I <sub>1</sub> | I <sub>2</sub> | I <sub>3</sub> | Compl. | O <sub>0</sub> | O <sub>1</sub> | O <sub>2</sub> | O <sub>3</sub> |
|----------------|----------------|----------------|----------------|--------|----------------|----------------|----------------|----------------|
| X              | X              | X              | X              | L      | I <sub>0</sub> | I <sub>1</sub> | I <sub>2</sub> | I <sub>3</sub> |
| L              | L              | L              | L              | H      | H              | L              | L              | H              |
| H              | L              | L              | L              | H      | L              | L              | L              | H              |
| L              | H              | L              | L              | H      | H              | H              | H              | L              |
| H              | H              | L              | L              | H      | L              | H              | H              | L              |
| L              | L              | H              | L              | H      | H              | L              | H              | L              |
| H              | L              | H              | L              | H      | L              | L              | H              | L              |
| L              | H              | H              | L              | H      | H              | H              | L              | L              |
| H              | H              | H              | L              | H      | L              | H              | L              | L              |
| L              | L              | L              | H              | H      | H              | L              | L              | L              |
| H              | L              | L              | H              | H      | L              | L              | L              | L              |

H = HIGH Voltage level  
L = LOW Voltage level  
X = Immaterial

This controlled nines complement circuit (Figure 5-40), using two gate packages, either generates the nines complement or transfers the BCD inputs through unchanged.